Homework 2: DNA Analysis
Adopted from UW CSE 160 homework
Due: July 6, 2016

Background

You will use, modify, and extend a program to compute the GC content of DNA data. The GC content of DNA is the percentage of nucleotides that are either G or C.

DNA can be thought of as a sequence of nucleotides. Each nucleotide is adenine, cytosine, guanine, or thymine. These are abbreviated as A, C, G, and T. A nucleotide is also called a nucleotide base, nitrogenous base, nucleobase, or just a base.

Biologists have multiple reasons to be interested in GC content:

- GC content can identify genes within the DNA, and can identify types of genes. Genes tend to have higher GC content than other parts of the DNA. Genes with longer coding regions have even higher GC content.
- Regions of DNA with higher GC content require higher temperatures for certain chemical reactions, such as when copying/duplicating the DNA.
- GC content can be used in determining classification of species.

If you are curious, Wikipedia has more information about GC content (https://en.wikipedia.org/wiki/GC-content). That reading is optional and is not required to complete this assignment.

Here are the first 8 lines of output from a particular sequencer:

```
@SOLEXA-1GA-2_2_FC30DNN:1:2:574:1722
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
+SOLEXA-1GA-2_2_FC30DNN:1:2:574:1722
hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh
@SOLEXA-1GA-2_2_FC30DNN:1:2:478:1745
GTGGGGGTGATGTCCACGATTACGCCGACCGGCTGG
+SOLEXA-1GA-2_2_FC30DNN:1:2:478:1745
hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh
```

The nucleotide data is in the second line, the sixth line, the tenth line, etc. Your program will not use the rest of the file, which provides information about the sequencer and the sequencing process that created the nucleotide data.

Problem 1: Obtain the files, add your name

Obtain the files you need by downloading the homework2.zip file (located in the Github repository). (this is a large download)

Unzip this zip file to create a homework2 directory/folder. You will do your work here. The homework2 directory/folder contains:

- dna_analysis.py, a partial Python program that you will complete
- answers.txt, a file where you will answer textual questions
- data, a directory. Which contains the data that you will process:

  - *.fastq files, which are output from DNA sequencers; this is the data that the program analyzes

- expected_output, a directory containing example runs of the final result of your dna_analysis.py program.

You will do your work by modifying two files (dna_analysis.py and answers.txt) and then submitting the modified versions. Add your name to the top of each of these files.

Each problem will ask you to make some changes to the program dna_analysis.py (or to write text in the answers.txt file, or both). When you do so, you will generally add to the program. Do not remove changes from earlier problems when you work on later problems; your final program should solve all the problems.

In either file, keep the number of characters on a particular line below 100, otherwise your files become hard to read. One technique to do this in Python is to break large equations into smaller ones by storing subexpressions in variables. Breaking things down into smaller computations can also help with debugging.
By the end of the assignment, we would like dna_analysis.py **to produce output of the exact form:**

```
GC-content: ___
AT-content: ___
G count: ___
C count: ___
A count: ___
T count: ___
Sum of G+C+A+T counts: ___
Total count: ___
Length of nucleotides: ___
AT/GC Ratio: ___
GC Classification: ___
```

Where ___ is replaced by values that you will calculate. Of course, the exact values in each category will vary depending on the input data that you are using. **We expect the formatting of your program output to exactly match this.**
Please validate your dna_analysis.py program's output using the Diff Checker
(https://www.diffchecker.com/diff) before submitting your assignment. You can compare your output to the files given in the expected_output directory of the homework2 files.

You will submit answers.txt as a **text file**. Plain text is the standard for communicating information among programmers, because it can be read on any computer without installing proprietary software. You can edit text files using IDLE or another text editor. If you use a word processor, then be sure to save the files as text.

Windows users should never use Notepad, as Notepad will mangle the line endings in the file; WordPad or Notepad++ are better alternatives. To cut and paste things from the Windows command prompt, right click at the top of the window to get a menu to show up, then select Edit-¿Mark. Now you can left click and drag the mouse to select text from anywhere in the command prompt window. When you have highlighted what you want, hit return or return to the menu and select Edit then Copy.

Problem 2: Complete the Program

I am purposely making the homework more difficult that intended. I want you to understand how to edit Python scripts provided to you and edit it for your own purposes. Furthermore, it will be extra practice with the material we have been discussing in class.

You will encounter functions, keywords or other code that you may not recognize at first. Don't fret! The dna_analysis.py file I will give you will contain lots of comments and feel free to ask me how they work or check the documentation in your leisure.

Complete the following function

`filename_to_string(dna_filename)`

This function converts the contents of dna_filename into one string of nucleotides. This function will return the string sequence of nucleotides.

This function will provide a cursory look on how file processing works. For now, I will give you the necessary information you will need. You will see this line in the provided in the script:

```
inputfile = open(dna_filename)
```

inputfile is a file object. The easiest analogy I can think of is to think the inputfile as a list of strings in which the elements are lines of the file. Suppose for an example we have example.txt that contains the following:

```
This is the first line of the file!
blablablablabla
This is the third line of the file!
```

You can call the open function: input = open('example.txt'). If I type out input[0], you will get:

```
"This is the first line of the file!\n"
```

Notice that there is new line escape character at the end of the string. If you call input[0].strip():

```
"This is the first line of the file!"
```

The escape character disappears. This function will be important when developing this function. Also, remember that you can loop over elements of a list. Thus, you can also loop over the elements of a file object where its elements are the lines.

More details about filename_to_string(dna_filename)

dna_filename is the parameter for this function. Look back at the example output by the DNA sequencer in the beginning of this assignment. The file object for this function will have the exact output. The nucleotide data starts on the 2nd line and then it will be every 4th line from there. Thus, the nucleotide sequence will be on 2nd, 6th, 10th, 14th, etc. Think carefully how we can achieve this pattern.

Concatenate all these lines into one string and return this sequence.

Complete the following function

`countGC(nucleotides)`

The goals of this function is to print the GC-content of an organism given a string of nucleotides. Return the total count of nucleotides. This function follows similarly to the list processing examples and exercises we have done in class.

Problem 3: Run the program

When writing programs that analyze data (or any other type of program) it is important to check the correctness of your programs. One way to do this is by comparing the output of your program to a computation done in some other way, such as by hand or by a different program. We have provided the test-small.fastq file for this purpose. This file is small enough that you can easily open it up in a text editor and calculate the GC content by hand. Then, run your program to verify that it provides the correct answer for this file.

You should be able to open up test-small.fastq and other input and output files in a text editor of your choice.

For this assignment you will run your program by **opening a shell or command prompt**. Follow the directions found on this page
(http://courses.cs.washington.edu/courses/cse160/15sp/shell-usage.html) which will teach you the basics of command-line navigation for your operating system. You should navigate to your homework2 directory, then type the following command:
On Windows:

```
python dna_analysis.py data\test-small.fastq
```

If you get a "can't open file 'dna_analysis.py'" error or a "No such file or directory" error, then perhaps you are not in your homework2 directory, or you mistyped the file name.

After you have confirmed that your program runs correctly on test-small.fastq, run your program on each of the 6 real sample_N.fastq files provided, by executing 6 commands such as:
if you are a Windows user:

```
python dna_analysis.py data\sample_1.fastq
```

Run your program on different data files by changing sample_1.fastq to a different file name in the commands above. Be patient, you are processing a lot of data, and it might take a minute or so to run. (If you are interested, sample_3.fastq and sample_4.fastq are from Streptococcus pneumoniae TIGR4, and sample_5.fastq is from Human herpesvirus 5.)

If you have already used the Output Comparison Tool (referenced at the bottom of the page), you might notice that some of your results are different than the example results. Don't worry about this this issue will be resolved in Problem 7.

Cut and paste the line of output produced by your program regarding GC-content when run on sample_1.fastq into your answers.txt file. (Note, this could take a minute or so to run.) For example, your answer might look like:

```
GC-content: 0.42900139393
```

(Note that this is not the answer you should expect to get, this is just an example of the format that your answer should be in.)

Problem 4: Remove some lines

a. In your program, comment out this line:

   ```
   gc_count = 0
   ```

   by prefixing it by the # character. Re-run the program, just as you did for Problem 3. In answers.txt, explain what happened, and why it happened.

b. Now, restore the line to its original state by removing the # that you added. What would happen if you commented out this line instead?

   ```
   nucleotides = filename_to_string(filename)
   ```

1 Problem 5: Compute AT content

Augment your program so that, in addition to computing and printing the GC ratio, it also computes and prints the AT content. The AT content is the percentage of nucleotides that are A or T.

Check your work by manually computing the AT content for file test-small.fastq, then comparing it to the output of running your program on test-small.fastq.
Run your program on sample_1.fastq. Cut-and-paste the relevant line of output into answers.txt.

Problem 6: Count nucleotides

Augment your program so that it also computes and prints the number of A nucleotides, the number of T nucleotides, the number of G nucleotides, and the number of C nucleotides.

Check your work by manually computing the results for file test-small.fastq, then comparing them to the output of running your program on test-small.fastq.

Run your program on sample_1.fastq. Cut-and-paste the relevant lines of output into answers.txt (the lines that indicate the G count, C count, A count, and T count).

Problem 7: Sanity-check the data

For each of the eleven .fastq files you have been given, calculate and print the following three quantities:

- the sum of: the A count, the C count, the G count, and the T count (store this in a new variable called sum_counts)
- the total_count variable (total number of nucleotides)
- the length of the nucleotides string variable. You can compute this with len(nucleotides).

In other words, compute the three quantities for test-small.fastq and then do the same for test-high-gc-1.fastq, etc.

For at least one .fastq file, at least one of these quantities will be different from the other two. In your answers.txt file, state which .fastq file(s) and which quantities differ. (If all three quantities are equal for each .fastq file, then your code contains a mistake.) In your answers.txt file, write a short paragraph that explains why these differ.

Explaining why (or debugging your code if all three quantities were the same in all .fastq files) might require you to do some detective work.

This exercise is meant to expose you to a situation you might encounter when processing a data file of your own (say on your Final project). When your program does not give the results you expect, there are two likely sources of the problem. One is that your program contains a bug! Check your code carefully to be sure you are calculating all values correctly. We will talk about testing in more detail later but for now, try walking through your code with a very small data set and calculating values by hand. A second source of unexpected results that is very common with data files is that there is something you were assuming about the contents of the data files that was an incorrect assumption. This could include things like assuming each line would contain a certain number of characters or words, or that all characters would be uppercase or lowercase, or that values might only be in a certain range. If you wrote your program assuming something about your data files that was not correct, your program may not give correct results.

To track down a wrong assumption about a data file, think about ways you can modify your program to help you determine what is happening. This could include having it print out values when they do not meet some asumption you are making about the file. You could also try just loading a data file into a text editor and examining it with your eyes to see if you see something you did not expect. (Although if you try this approach we strongly suggest that you start with the smallest data file for which the three quantities are not all the same.) Another approach would be to modify your program, or create a new program, to compute the three quantities for each line of a data file separately (as opposed to for the file as a whole as you have been doing): if the quantities differ for an entire file, then they must differ for at least one specific line in that file. Examining that/those line(s) will help you understand the problem.

If all of the three quantities that you measured in problem 7 are the same, then it would not matter which one you used in the denominator when computing the GC content. However, you saw that the three quantities are not

all the same. In answers.txt, state which of these quantities should be used in the denominator and which should not, and why.

If your program incorrectly computed the GC content (which should be equal to (G+C)/(A+C+G+T)), then state that fact in your answers.txt file. Then, go back and correct your program, and also update any incorrect answers elsewhere in your answers.txt file. It is fine to change the code we provided you

**If you are unsure if you are calculating things correctly, now would be a good time to validate your dna_analysis.py program's output using the Diff Checker. You can compare your output to the files given in the expected_output directory of the homework2 files. You have not yet completed the assignment, so your output will not be identical. But things like GC-content, AT-content and individual counts should be identical. You will produce the last two lines of output in the expected_output files in Problem 8 and Problem 9 below.

Problem 8: Compute the AT/GC ratio

Sometimes biologists use the AT/GC ratio, defined as (A+T)/(G+C), rather than the GC-content, which is defined as (G+C)/(A+C+G+T).

Modify your program so that it also computes the AT/GC ratio.

Check your work by manually computing the results for file test-small.fastq. Compare them to the output of running your program on test-small.fastq.

Run your program on sample_1.fastq. Cut-and-paste the relevant lines of output into answers.txt (the line that indicates the AT/GC ratio).

Problem 9: Categorize Organisms

The GC content can be used to categorize microorganisms. Modify your program to print out a classification of the organism described in the data file given using these classifications:

If the GC content is above 60%, the organism is considered high GC content. If the GC content is below 40%, the organism is considered low GC content. Otherwise, the organism is considered moderate GC content.

Biologists can use GC content for classifying species, for determining the melting temperature of the DNA (useful for both ecology and experimentation, for example PCR (https://en.wikipedia.org/wiki/Polymerase_chain_reaction) is more difficult on organisms with high GC content), and for other purposes. Here are some examples:

The GC content of Streptomyces coelicolor A3(2) is 72%.
The GC content of Yeast (Saccharomyces cerevisiae) is 38%.
The GC content of Thale Cress (Arabidopsis thaliana) is 36%.
The GC content of Plasmodium falciparum is 20%.

Again, test that your program works on some data files with known outputs. The test-small.fastq file has low GC content. We have provided four other test files, whose names explain their GC content: test-moderate-gc-1.fastq, test-moderate-gc-2.fastq, test-high-gc-1.fastq, test-high-gc-2.fastq.

After your program works for all the test files, run it on sample_1.fastq. Cut-and-paste just the relevant line of output from your program into answers.txt.

Submit your work!

I recommend doing a quick search on this web page for answers.txt to confirm that each place we asked you to answer a question, you have answered it.

Email me the two files! Good luck!